Digenes: genetic algorithms to discover conjectures about directed and undirected graphs

Romain Absil[*,†] Hadrien Mélot[*,‡]

May 1, 2013

Abstract. We present Digenes, a new discovery system that aims to help researchers in graph theory. While its main task is to find extremal graphs for a given (function of) invariants, it also provides some basic support in proof conception. This has already been proved to be very useful to find new conjectures since the AutoGraphiX system of Caporossi and Hansen [8]. However, unlike existing systems, Digenes can be used both with directed or undirected graphs. In this paper, we present the principles and functionality of Digenes, describe the genetic algorithms that have been designed to achieve them, and give some computational results and open questions. This do arise some interesting questions regarding genetic algorithms design particular to this field, such as crossover definition.

Keywords: Discovery system; genetic algorithm; directed graph, Digenes.

1 Introduction

During the last three decades, several software systems have been dedicated to the task of helping the graph theorist in the process of discovery. This computer aided support is currently much developed to obtain new conjectures or to search for counterexamples. Less efforts (or successes) have been made to assist the writing of proofs and, to the best of our knowledge, there exists no discovery system dedicated to *directed* graphs, all theses systems dealing only with undirected graphs. After settling some notations in Section 2, we briefly survey existing discovery systems in Section 3.

In this paper, we introduce the new system Digenes that can deal both with undirected *and* directed graphs. Its principles and functionality are described in Section 4. The main goal of Digenes is to find graphs that are extremal for a given objective function of graph invariants. Moreover, we explain how this system can be used to make conjectures; find graphs satisfying some given constraints; search for counterexamples of a given conjecture; and check graph transformations that can be used in proofs.

[*]Algorithms Lab, Université de Mons, Place du parc 20, B-7000 Mons, Belgium.
[†]FRIA grant holder.
[‡]Corresponding author. E-mail: romain.absil@umons.ac.be.

In Section 5, we describe the algorithms and choices we made to develop Digenes. These are based on genetic algorithms and do arise interesting questions about graphs encodings and operators (crossovers and mutations) specific to the task of finding extremal graphs.

We validate our algorithms in Section 6 with computational results. Finally, we end this paper by drawing concluding remarks and pointing out challenging open questions.

2 Notations

This section is devoted to basic definitions and notations used throughout the paper. We assume the reader to be familiar with usual notions of graph theory and we refer to the books of Diestel [12] for general graph theory and Bang-Jensen and Gutin [6] for more details about directed graphs (*digraphs*).

Let $G = (V, A)$ be a simple digraph of order $n(G) = |V|$ and size $m(G) = |A|$. We denote by $\mathcal{G}_n$ the space of all simple non isomorphic digraphs of order n, and $\widehat{\mathcal{G}_n}$ the space of all simple non isomorphic undirected graphs of order n. A (graph) *invariant* is a numerical value preserved by isomorphism such as chromatic number, independence number, diameter and so on. We also provide some invariants definitions used later.

Recall that the *diameter* $D(G)$ of a graph G is the maximum distance between any pair of its vertices. The *average distance* $\mu(G)$ of a graph G is the arithmetic average of the lengths of all shortest paths of G. In the case of digraphs, G must be strongly connected in order to have a finite diameter or average distance. In an undirected graph $G = (V, E)$, the *imbalance* of an edge $\{i, j\}$ is defined as $|d_i - d_j|$, where d_i is the degree of a vertex $i \in V$, and the *irregularity* $irr(G)$ of G as the sum of imbalances of its edges, as firstly stated by Albertson [1].

We note $\mathcal{C}_n$ and $\mathcal{K}_n$ the cycle and the complete graph of order n, respectively. We note $KS_{k,l}$ the undirected complete split graph made by connecting all vertices of a complete graph of order k to all vertices of an empty graph of order l.

Let f be an objective function defined by an invariant (or an arithmetic formula depending on some graph invariant – that is obviously also an invariant), we will consider combinatorial optimisation problems of the form

$$\max_{G \in \mathcal{G}_n} f(G), \tag{1}$$

$$\max_{G \in \widehat{\mathcal{G}_n}} f(G). \tag{2}$$

where $\mathcal{H}_n \subseteq \mathcal{G}_n$ is a class of *digraphs*, and $\widehat{\mathcal{H}_n} \subseteq \widehat{\mathcal{G}_n}$ is a class of *undirected* graphs. For example, $\mathcal{H}_n$ can be the class of strongly connected digraphs of order n and $f(G)$ can be defined by $D(G)$.

To simplify presentation, we will only consider maximisation problems since a minimisation problem can easily be translated into a maximisation problem by multiplying the objective function by -1.

3 Existing graph theoretical discovery systems

Research in mathematics is increasingly aided by computers and graph theory clearly makes no exception since the proof of the 4-color theorem [3–5,34]. For example, it is frequent to use graph generators (such as geng of McKay [29,30]) to confront a conjecture to a large set of (small) examples. This enumerative approach can lead to a counter example, or if it is not the case, reinforce the belief that a tested conjecture is true. Also, some conjecture making systems, such GraPHedron [31], use an enumerative step in their underlying processes.

Other systems allow to interactively explore properties of a selected set of graphs, such as invariant's values. It is for instance the case of the pioneering system Graph [9–11], its sequel newGraph [35] and Grinvin [33] (the later one specifically designed for pedagogical purposes).

The Graffiti system [14–18] uses a specific process to maintain a database of counter-examples and conjectures. It has led to hundred conjectures, some of which have attracted much attention.

Finally, we mention AutoGraphiX (AGX) [2,8]. The main idea of AGX is to write undirected graph theory problems of type (2) as combinatorial optimisation problems and then use an heuristic, based on a Variable Neighbourhood search (VNS) [27,32], to approximate an optimal solution.

In addition to this very short survey, the next section shows other ways that can be used to assist the writing of conjectures and proofs. The interested reader can also find more detailed surveys on conjecture-making systems in the papers of Hansen *et al.* [23,25]. We observe that all the surveyed systems only deal with *undirected* graphs.

4 Principles and functionality of Digenes

As previously introduced, existing systems offer various features, although they never deal with directed graphs. In this section, we present a new system, called Digenes[1], its basic principles and core features.

We note that while exact methods, such as enumeration used for instance in GraPHedron, might be suited to solve undirected graphs problems of type (2), they are most likely inapplicable in the directed case. Indeed, the number of non isomorphic directed graphs[2] increases far more quickly than the undirected one[3]. For instance, there are 12005168 undirected graphs of order 10, while there are 341260431952972580352 (giving for this order a factor of about 10^{14}). This growth motivates the use of metaheuristics for these problems, as they allow to find assumed good solutions in reasonable time. As far as we know, VNS is the only metaheuristic that has been ever used to solve such problems, within AGX.

[1] Digenes is actually a contraction of the words "Directed" for directed graphs and "Genetic" for genetic algorithms. Digenes source code is available by request from the corresponding author.

[2] N. J. A. Sloan - OEIS Foundation - `www.oeis.org`, Sequence A000273 - 6/12/2012.

[3] N. J. A. Sloan - OEIS Foundation - `www.oeis.org`, Sequence A000088 - 6/12/2012.

Therefore, a natural question is to wonder how another metaheuristic will succeed for this type of problems, and to compare the quality of the approximated solutions or the time elapsed to find them. We give some answers to this question in this paper since Digenes uses genetic algorithms (GA) [36] we specialise to study extremal graph theory.

A point motivating this choice is that GA are a population metaheuristic. If for some invariant the maximum is actually reached on some class of graphs, a single solution heuristic, such as local search, will only output a single graph while we have to guess the non uniqueness of this solution. On the other hand, population heuristics will output a set of solutions, usually with several assumed extremal solutions within. Also, we note that many algorithmic methods running time grows in particular with optimisation space size. Genetic algorithms do not, their worst case time complexity only growing with population size, linearly[4].

As previously stated by Hansen and Caporossi [8], problems modelled as (1) and (2) might seem simple (and restrictive), however, many graph theory problems can be modelled using this formalism. More concretely, Digenes offers the following features that can be used for discovery, both at the levels of conjectures and proofs :

1. find extremal graphs for some invariant,
2. find counterexamples to conjectures or graphs satisfying a given set of constraints,
3. check graph transformations.

We detail these features in the following subsections. Note that since Digenes works with a metaheuristic, all of these features are to be understood as "assumed optimal". Indeed, a found maximum may well be just local rather than global in the case of extremal graph finding. In the same way, just because no counterexample has been found to a conjecture does not mean it is true.

As matter of fact, we introduce each of these features by a simple example, and explain then how to generalise it for any similar problem.

4.1 Extremal graphs finding and constraint handling

It is often considered research in extremal graph theory started in 1941 with the result of Turàn [37], who wondered what was simple yet common question in extremal graph theory is to wonder, given a specific invariant f, what is the maximal value that f can reach on some graph G, possibly subject to constraints. We illustrate by a simple example how Digenes handles the matter on Problem 1, along with an easy constraint handling.

Problem 1 (Directed diameter)**.** Let G a strongly connected digraph of order n, what are the graphs maximising $D(G)$?

[4]It actually also grows with the time complexity of its inner components, but so do other metaheuristics.

For this problem, one can easily show that

$$D(G) \leqslant n - 1,$$

with equality if and only if G has an hamiltonian path with no forward arc.

A first naive way to model this problem is to launch the algorithm on the problem "$\max D(G)$" with no constraints. However, diameter can only be computed on strongly connected graphs[5]. On the other hand, restricting Digenes to only work with strongly connected graphs is unwise since non strongly connected graphs might provide good crossover or mutations scheme. Usually, literature recommends to penalise unfeasible individuals in the objective function. Considering these improvements, we can now model Problem 1 as follows :

$$\max_{G \in \mathcal{G}_n} \frac{1}{k(G)} \cdot D(G'), \tag{3}$$

where $k(G)$ is the number of strongly connected components of G, and G' is a graph constructed from G by adding a minimum number of arcs to make it strongly connected.

Of course, there are several ways to add such arcs, with possibly different diameter. This allow us to define a new local optimisation problem in which it might be useful to construct G' maximising diameter.

This way, not strongly connected graphs will have a lower objective value, and the "less strongly connected" they are, the less their objective value. When launched on this problem for $n = 10$, Digenes outputs a population, a sample of which is illustrated in Figure 1. We notice these sample graphs are actually extremal graphs described in Problem 1.

We can generalise this approach on any function f of invariants. The first step is to handle constraints, then make any graph feasible with a minimum modifications. In our example, we had to ensure the graph was strongly connected by adding a minimum number of arcs (or solve a optimisation subproblem). Finally, assuming Digenes found graphs belonging to some class $\mathcal{M}$ and maximising f reaching a maximum of value m, it automatically suggest the following conjecture :

Conjecture 2. Let $G \in \mathcal{G}_n$, we have

$$f(G) \leqslant m$$

with equality if and only if $G \in \mathcal{M}$.

Of course the definition of $\mathcal{M}$ will depend on some generalisation of what Digenes actually found, for instance if all extremal found graphs are trees, we will assume $\mathcal{M}$ is actually the set of all trees and not only the set the found ones. On the other hand, if

[5] Actually we are only interested in finite diameter, explaining why we only consider strongly connected graphs

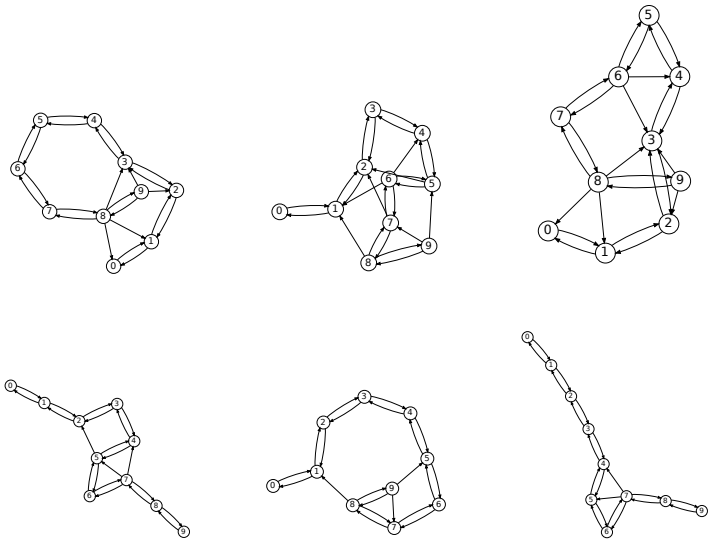

Figure 1: Sample of output population of Problem 1.

there is a unique extremal graph instead of a class, we will assume equality holds if and only if G is isomorphic to this graph rather than belonging to some class. For instance, it is well known that diameter is maximum on undirected graphs only on the path $\mathcal{P}_n$. For this problem, Digenes actually outputs a population consisting f many isomorphic copies of $\mathcal{P}_n$. We will see in Section 6 examples where only one graph is extremal.

4.2 Finding graph satisfying constraints or counterexamples

Given a set of constraints of type $\mathcal{I}(G) \leqslant \mathcal{I}'(G)$ or $\mathcal{I}(G) = \mathcal{I}'(G)$, where $\mathcal{I}$ and $\mathcal{I}'$ are both graph invariants, one can simply wonder if there exists graphs satisfying these conditions. Moreover, if there are, how can we characterise these graphs ? It is relevant consider such constraints since Hansen *et al.* [23] stated that most of conjectures and results in graph theory are equalities or inequalities between graph invariants.

The following simple example illustrates how we can solve such types of problems.

Problem 3. Is there any undirected graph of order n and size m such as $m = n - 1$?
It is trivial there are, however we are more interested in characterise these graphs. In that case, one can easily show that all undirected trees, and only them, match the property.

We can simply model this problem in Digenes by "$\max -|m - n + 1|$". If a graph have more or less edges than vertices, it will have a negative objective value. Otherwise, if the constraint is respected, it has a zero objective value.

We can generalise the handling of multiple constraints of type $\mathcal{I}_k(G) = \mathcal{I}'_k(G)$ simply by adding the terms $-|\mathcal{I}_k(G) - \mathcal{I}'_k(G)|$ to the objective function. We handle inequality constraints in a similar way.

This simple tool can also be used in order to find counterexamples to check, reinforce or reject conjectures. For instance, if we have a conjecture of the same type as Conjecture 2, we simply have to maximise $f(G) - m$. If a graph G is found with $f(G) - m > 0$, the conjecture is rejected.

4.3 Help for proofs and transformation validation

A common problem in extremal graph theory is to prove some graph G^* to be optimal for some invariant $\mathcal{I}$, *i.e.,* $\forall G \in \mathcal{G}_n, \mathcal{I}(G) \leqslant \mathcal{I}(G^*)$. A simple yet powerful approach to solve this problem is to use transformation proof. Intuitively, the idea is to find a finite sequence of k graph transformations $\mathcal{T}_i$ that will eventually end up on G^* by always increasing the value of $\mathcal{I}$. More formally, at each step and for any graph G and some graph transformation $\mathcal{T}_i$ (for $1 \leqslant i \leqslant k$), we want to ensure that

$$\mathcal{I}(G) < \mathcal{I}(T_i(G)). \tag{4}$$

For instance, Hansen *et al.* [24] used 5 graph transformations increasing $\mu(G)$ while keeping $F(G)$ unchanged, where $F(G)$ is the order of a maximal induced forest.

A key question in transformation proof is then, given a graph transformation $\mathcal{T}_i$, to wonder if Property 4 holds. If it does, we say that $\mathcal{T}_i$ is *valid*. Digenes provides help in this sense.

Indeed, while the system will not suggest a valid transformation (that is, a working one), given a transformation, Digenes will prove it invalid by automatically finding a counterexample, or will conjecture it to be valid when unable to find such a counterexample.

Again, just like in the previous sections, simple use of combinatorial modelling with the problem "$\max \mathcal{I}(G) - \mathcal{I}(\mathcal{T}(G))$" solves the problem. At the end of the optimisation process, if some G is found among the population with a positive value of objective function, then it is a counterexample and the transformation is not valid. Moreover, counterexamples to an invalid transformation are still useful since they can serve as basis for the construction of another transformation.

On the other hand, a common practice in graph transformation is to wonder whether some properties are kept under graph transformation. More concretely, given a property P and a transformation $\mathcal{T}$, one can simply ask the following question : if P holds on some graph G, does it always holds on $\mathcal{T}(G)$? This is a second main feature in Digenes regarding transformation validation.

The system proceeds basically in the same way, by modelling the problem under a combinatorial optimisation problem. Then the system is able to reject the transformation if a counterexample is found during the optimisation process, or conjecture it valid otherwise. This feature is illustrated in the following example.

Problem 4. Let $G \in \mathcal{G}_n$, and $P(G)$ the following property : $\mathcal{I}_1(G) \leqslant \mathcal{I}_2(G)$, where $\mathcal{I}_1$ and $\mathcal{I}_2$ are two graph invariants. Given some graph transformation $\mathcal{T}$, we want to check whether

$$\mathcal{I}_1(G) \leqslant \mathcal{I}_2(G) \Rightarrow \mathcal{I}_1(\mathcal{T}(G)) \leqslant \mathcal{I}_2(\mathcal{T}(G))$$

holds on any graph G. This question can be simply modelled by the following problem :

$$\begin{array}{l} \max f(G) \\ \text{where } f(G) = \begin{cases} \max \mathcal{I}_1(\mathcal{T}(G)) - \mathcal{I}_2(\mathcal{T}(G)) & \text{if } \mathcal{I}_1(G) \leqslant \mathcal{I}_2(G), \\ \max \dfrac{\mathcal{I}_1(\mathcal{T}(G)) - \mathcal{I}_2(\mathcal{T}(G))}{\mathcal{I}_1(G) - \mathcal{I}_2(G)} & \text{otherwise.} \end{cases} \end{array}$$

As before, graphs with positive value of objective function are counterexamples, and penalising non feasible graphs in the objective function improve genetic algorithm effectiveness.

In practice, of course, the objective function design widely depends on the property you want to know if it's kept under graph transformation. Moreover, various types of properties can be modelled in the way it is in Problem 4, thus allowing to study these type of questions.

5 A genetic algorithm for extremal graphs

In this section, we describe exactly what genetic algorithm components are implemented in Digenes, as well as how they are. We hereby assume the reader familiar with general genetic algorithm concepts [36]. We start with several initial population generators, follow with crossover and selection operators, and finish with mutations.

Initial population generators

A first and core component of genetic algorithms is the initial population from which you start, an assumed "well spread" sample of the studied optimisation space. There are several standard generations routines [36][pp. 193-198], we present here four of those available in Digenes, namely

- random generator,
- random degree sequence generator,
- sized block generator,
- House of Graphs (HoG) [7].

A first idea in order to generate random graphs is to generate random encodings of graphs. Random generation and random degree sequence generation work in that way. The first one randomly generates binary adjacency matrices, the second one random degree sequences, and then map that coding to a given graph. In random generation, each arc has then a given probability to be in the graph, in random degree sequence generation, each vertex has a randomly generated in and out-degree. We use the Havel-Hakimi algorithm [22, 28] to map degree sequences to directed graphs. If the mapping algorithm fails, the degree sequence is dropped and a new one is generated.

In sized block generation, the basic idea is to spread graph size. That is, at order n, for each possible size $m \in [0, n(n-1)]$, we generate k_m graphs. In order to respect graph size distribution as much as possible and in polynomial time, we use their known repartition for small orders[6], and extrapolate this repartition for bigger graphs. Then, we only have to generate graphs in blocks of fixed size k_m.

Finally, HoG [7] is a small database of "interesting" graphs, for instance complete graphs, cycles, stars, snarks, trees, etc. The idea is then to explicitly add some of these graphs in the initial population, along with some randomly generated graphs (with one of the previous schemes). We will see in Section 6 that this simple feature might vastly increase the algorithm performances.

Selection operators

Selection methods are a core concept in genetic algorithm, since they select which individuals mate and who survives in the next generation. Digenes implements several of standard operators existing in literature [36, pp. 205-221].

More particularly, Digenes offers four main selection methods : roulette wheel selection, stochastic universal sampling, tournament selection and direct elimination.

Intuitively, in roulette wheel strategy, each individual has a probability directly proportional to its fitness to be selected. Stochastic universal sampling proceeds in a similar way, with lower bias. In tournament selection, a number k of individuals is randomly chosen from the population, the best of these ones is output. This procedure is repeated as many times as needed. We note that adjusting wisely the value of the parameter k usually have a significant impact on the algorithm convergence [21]. Finally, in direct elimination, individuals are placed randomly on a direct elimination grid, and individuals are selected after computing the winners.

Crossover schemes

Crossover operators define most of the evolution behaviour of a genetic algorithm, *i.e.,* how to mate parents and output siblings. Digenes allows the use of various types of

[6]N. J. A. Sloan - OEIS Foundation - `www.oeis.org`, Sequence A052283 - 06/12/2012.

standard crossovers operators, like k-Points crossover, as well as some ones more particular to graphs, like Align-Greedy.

In k-points crossover, parent graphs are encoded under integer array format, then k cross-points are randomly chosen. Offsprings are computed by alternatively picking parts of the two parents encodings. For this purpose, two types of encodings are provided, although they basically work the same way. The first one simply put the adjacency matrix of a directed graph into a one dimensional array. The second one does the exact same thing, but sorts vertices by decreasing in and out-degrees. For undirected graphs, only the upper triangular part of the adjacency matrix is encoded.

While this idea is simple, we note a slight problem regarding graph encoding : two isomorphic graphs can have different encodings, that is, the encoding operator is sensitive to vertex labelling. More concretely, if two isomorphic graphs mate, they might give birth to two non isomorphic children. Since isomorphism problem solving is difficult [20], a basic idea is to use heuristics to solve this disparity while avoiding long computations.

A simple yet powerful approach is simply to use local search on the graph adjacency matrix. More formally, given two matrices A and B, the algorithm swaps rows and columns i and j in B if it increases the number of matching entries between A and B, and repeat that process as many times as possible. When such a permutation cannot be found, we said that the two corresponding graphs are *aligned*.

A first simple crossover scheme is then to apply k-points crossover on two aligned graphs. We note that crossover Align-KPoints. Another idea is, with aligned graph G_1 and G_2, to alternatively pick good subgraphs from G_1 and G_2 to build children. Of course, the way subgraphs are picked depends on the studied problem. For instance, we can use a greedy heuristic with the underlying fitness to choose "the best" way to pick a vertex from one of these graphs. We call this crossover scheme Align-Greedy

Mutations

Basically, a mutation is a function modifying a graph in order to promote diversity, for instance by adding, removing or moving some of its edges. Each generation, some individuals mutate (with an assumed low probability) to produce new, different individuals.

Digenes offers various general (small) graph mutation schemes, inspired from local search moves we can for instance find in AGX [8] and that we particularise for directed graphs. Figure 2 illustrates some of these mutations. In order to reinforce the mutation factor, it is sometimes useful to chain some of these schemes multiple times in the same mutation process.

6 Computational results

In the following section, we present some computational results in order to validate Digenes and its underlying algorithms. For this purpose, we consider Problem 1, along with the

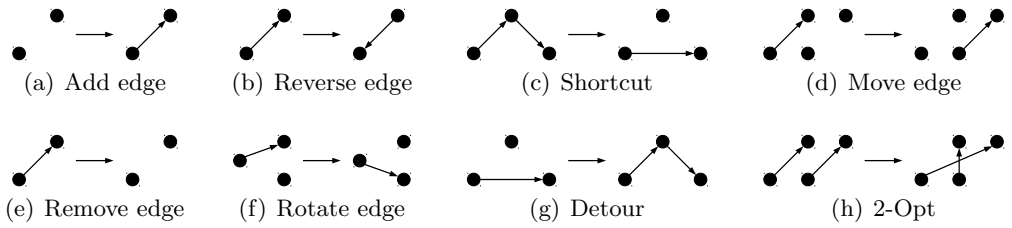

Figure 2: Illustration of graph mutations

two following ones.

Problem 5 (Directed average distance)**.** Let G a strongly connected digraph of order n, what are the graphs maximising $\mu(G)$?
For this problem, Doyle and Graver [13] proved that

$$\mu(G) \leqslant \frac{n}{2},$$

with equality if and only if $G \simeq \mathcal{C}_n$.

Problem 6 (Undirected irregularity)**.** Let G an undirected graph of order n, what are the graphs maximising $irreg(G)$?
Albertson [1] showed that

$$irreg(G) \leqslant \frac{4n^3}{27},$$

although this bound is not tight. Moreover, Hansen and Mélot [26] proved that irregularity is maximal if and only if

$$G \simeq \begin{cases} KS_{\frac{n-2}{2}}\ ,\ \frac{n+2}{2} & \text{if } n \text{ is even,} \\ KS_{\frac{n-3}{2}}\ ,\ \frac{n+3}{2} & \text{otherwise.} \end{cases}$$

Table 1 denotes statistics regarding these three problems. Each problem has been launched a hundred times, during 20000 generations, with stochastic universal sampling selection both for parents and survivors selection, Align-2Point crossover and an initial population generated in blocks of fixed size along with some graphs picked from HoG.

For each of the subtables, first column denotes the order of computed graphs, the second one being the *raw success* (R. S.) of the algorithm, that is, the number of times the algorithm actually found the optimum, while the third one denotes the *amortised success* (A. S.), *i.e.,* the average of ratios between the maximum found fitness and the optimum fitness. The fourth column gives the average time (t.(s)), in CPU seconds[7], to run one test, and the last one the average generation number (t.(gen)) on which the optimum is firstly found, when it is found.

[7]Output from an Intel © Core ™ i5 420M @ 2.53GHz.

n	Directed diameter				Directed average distance				Undirected irregularity			
	R. S.	A. S.	t.(s)	t.(gen)	R. S.	A. S.	t.(s)	t.(gen)	R. S.	A. S.	t.(s)	t.(gen)
5	100	1.0	15.3	17.8	100	1.0	16.5	48.3	100	1.0	6.3	16.3
10	100	1.0	26.3	85.4	100	1.0	34.6	99.1	100	1.0	12.1	76.2
15	100	1.0	54.4	260.5	100	1.0	51.4	296.6	100	1.0	36.7	198.9
20	100	1.0	88.7	784.2	98	0.99	94.1	983.5	100	1.0	73.5	634.7
25	100	1.0	146.6	1812.8	91	0.94	158.6	2645.2	100	1.0	126.2	1227.1
30	100	1.0	231.2	4769.1	82	0.88	212.9	6839.9	99	0.99	157.6	2531.5
35	92	0.95	273.7	8543.7	69	0.77	293.7	9671.5	87	0.93	201.9	4672.9
40	85	0.90	401.3	12391.4	53	0.62	421.2	16428.6	79	0.84	248.5	8740.1

Table 1: Output statistics on directed diameter and average distance, and undirected irregularity.

As we could have expected, system "efficiency", *i.e.,* the average successes and execution times, widely depends on the studied problem. Moreover, the fact that irregularity is an invariant of lesser worst case complexity, and that this problem is stated on undirected graphs can explain the improved running time for this invariant.

Additionally, from this first table, we can see that Digenes is scalable with graph order. Indeed, while with average distance the raw success decreases a lot past a threshold, the amortised success remains acceptable. This suggests that, when no optimal solution can be found, it is still possible to find an individual with a fitness close to the optimal solution.

However, we have to keep in mind that these statistics could be output since we know optimal values for each underlying problem. When it is not the case, *i.e.,* when Digenes is actually used to study new problems or conjectures, it would be wise to restrain the system to graphs of smaller orders, as we have no optimality guarantee. Moreover, in every case, a generic crossover operator such as Align-2Point was enough to find optimal values. Literature shows however that sometime, *e.g.,* with graph colouring [19], design a custom crossover dedicated to the studied problem is necessary.

The following table show parameters variations effects, that is, how the genetic algorithm behaves when changing some of its core components, like initial population, crossover and selection operators. Concretely, Table 2 illustrates this behaviour for average distance, computed for graphs of order 15. Each table entry denotes amortised success for the underlying parameters, computed over 100 tests.

We observe that a simply randomly generated population is usually less effective than a population generated in blocks of fixed size. This might mean that size matters more than isomorphism distribution. Moreover, we noticed with further experiments that forcing at least one graph of each possible size to be generated improves amortised success by about 4%. These are the results detailed in Table 2. We also note that the HoG improvement slightly increases the algorithm efficiency.

Another benefit of this improvement, for which statistics are not detailed here, is that it vastly decreases the average number of generations needed to find optimums the first time. More particularly, we observe improvements from about more than 4000 needed

Selection	Crossover	Initial population			
		Random	Deg. Seq.	Size block	HoG
Direct Elim.	1-Point	0.81	0.71	0.84	0.85
	2-Point	0.84	0.78	0.90	0.90
	Align-2Point	0.87	0.79	0.91	0.94
	Align-Greedy	0.88	0.81	0.90	0.94
Roulette Wheel	1-Point	0.83	0.74	0.91	0.91
	2-Point	0.89	0.81	0.93	0.95
	Align-2Point	0.92	0.88	0.98	1.0
	Align-Greedy	0.95	0.87	0.97	0.98
Stoch. Un. S.	1-Point	0.81	0.76	0.91	0.91
	2-Point	0.91	0.84	0.95	0.96
	Align-2Point	0.95	0.89	1.0	1.0
	Align-Greedy	0.95	0.86	1.0	1.0
Tournament	1-Point	0.85	0.73	0.88	0.89
	2-Point	0.88	0.82	0.91	0.93
	Align-2Point	0.90	0.83	0.93	0.95
	Align-Greedy	0.89	0.86	0.94	0.96

Table 2: Algorithm behaviour on operator switching for diameter.

generations to less than 1500, considering the underlying problem.

On the other hand, generating graph from spreading over a coding in degree sequence is the least efficient initial population among the three other ones. So far, we assume that the naive filtering routine described in Section 5 is inappropriate to generate a "good" initial population. Future work trying to improve this random generation could increase the effectiveness of this generator.

Regarding crossover operators, we observe that 2-Point crossovers generally provide better results than 1-Point crossover. An important point however is that aligning graphs and then perform a 2-Point crossover over their encodings always improves the algorithm efficiency. However, for average distance, guiding the algorithm by choosing good subgraphs for the children does not seem to improve the algorithm. This could be explained by the fact that handling strong connectivity constraints in subgraphs penalises greatly optimisation, since these subgraphs will most likely not be strongly connected, especially smaller ones. As debated before, in problems such as graph colouring, this approach might however be extremely efficient.

Finally, roulette wheel selection seems to be more efficient than direct elimination. This could be explained by the fact that direct elimination is more (maybe too much) elitist in the way that, when using this routine, maximum is always selected while minimum is always dropped. Regarding the algorithm diversity and convergence, this can be a drawback in case of "too deep" local optimum. Moreover, reducing bias in universal stochastic sampling slightly improves the algorithm efficiency. At last, we note that tournament selection usually provides better results than direct elimination, although it stays less efficient than roulette wheel selection.

Similar tables for irregularity and diameter have not been provided since the system always find the optimal solution. More particularly, only switching initial population for these two invariants have a notable effect, and only on the time (in generations) needed to find the maximum the first time, in a range from 150 to 800 generations, not on the overall success.

7 Concluding remarks and open questions

We considered a particular type of optimisation problem using graphs as *feasible solutions* and not as instances, as it is often the case. They are most useful in extremal graph theory where they allow computer use in order to study problems and discover new results, such as conjectures, counterexamples, proof leads, etc. Numerous results output by AutoGraphiX [8] justify this approach.

In this paper, we showed it was possible to design *genetic algorithms* to solve these extremal graphs problems. Moreover, we described the system Digenes implementing such algorithms and that is the first system to deal with directed graphs (as well as undirected ones). This system allows to find not only extremal graphs but also graphs satisfying given constraints, counterexamples and to check a graph transformation operator validity. Computational results show that this approach is working and relatively efficient.

It is interesting to note that considered optimisation problems arise several fundamental questions and open problems regarding the use of genetic algorithms when individuals are (directed) graphs. These questions are not trivial since depend most of the time on the underlying studied problem, for instance on invariants defining the objective function or on graph constraints. We list here some of these questions :

- How to efficiently design objective function to deal with *soft* constraints sensitive to crossover and mutation operators ?
- How to define crossover and mutation operators preserving *hard* constraints ?
- How to define crossover operators handling heredity relevantly ?
- How to design particular operators or encoding less sensitive to the studied problem ?
- More generally, is it possible to design a genetic algorithm which is the most generic possible for any type of graph finding problem ?
- Would other metaheuristics be more suitable for this type of problems, other than VNS (AutoGraphiX) and genetic algorithms (Digenes) ?

We gave some elements of answers to some of these questions although there are still many leads to follow, for instance regarding hybrids auto-adaptive algorithms which would automatically determine the best fitting operators (among available ones) for any given problem.

Acknowledgments

Romain Absil has a Phd funded by the FRIA. We also thank Alain Hertz, for useful discussions about crossover design we use in this paper.

References

[1] Albertson, M.O. The Irregularity of a Graph. *Ars Combinatoria 46* (1997), 219 – 225.

[2] Aouchiche, M., Caporossi, G., Hansen, P., and Laffay, M. AutoGraphiX: a survey. *Electronic Notes in Discrete Mathematics 22* (2005), 515 – 520.

[3] Appel, K., and Haken, W. Every Planar Map is Four Colorable. Part I. Discharging. *Illinois Journal of Mathematics 21* (1977), 429 – 490.

[4] Appel, K., and Haken, W. Every Planar Map is Four Colorable. Part II. Reducibility. *Illinois Journal of Mathematics 21* (1977), 491 – 567.

[5] Appel, K., and Haken, W. Every Planar Map is Four Colorable. *Contemporary Mathematics 98* (1989), 1 – 741.

[6] Bang-Jensen, J.., and Gutin, G., Eds. *Digraphs: Theory, Algorithms and Applications*. Springer, New York, 2001.

[7] Brinkmann, G., Goedgebeur, J., Mélot, H., and Coolsaet, K. House of graphs: a database of interesting graphs. *Discrete Applied Mathematics 161* (2013), 311–314.

[8] Caporossi, G., and Hansen, P. Variable Neighborhood Search for Extremal Graphs 1. The AutoGraphiX System. *Discrete Math. 212* (2000), 29 – 44.

[9] Cvetković, D., and Kraus, L. "Graph" an Expert System for the Classification and Extension of the Knowledge in the Field of Graph Theory, User's Manual. Elektrotehn. Fak., Beograd, 1983.

[10] Cvetković, D., Kraus, L., and Simić, S. Discussing Graph Theory with a Computer, I: Implementation of Graph Theoretic Algorithms. *Univ. Beograd Publ. Elektrotehn. Fak, Ser. Mat. Fiz. No. 716 – No. 734* (1981), 100 – 104.

[11] Cvetković, D., and Simić, S. Graph Theoretical Results Obtained by the Support of the Expert System "Graph". *Académie Serbe des Sciences et des Arts. Classe des Sciences MathÃ©matiques et Naturelles 19* (1994), 19 – 41.

[12] Diestel, R. *Graph Theory*, second edition ed. Springer-Verlag, 2000.

[13] Doyle, J. K., and Graver, J. E. Mean distance in a directed graph. *Environment and planning B 5* (1978), 19–25.

[14] Fajtlowicz, S. Written on the Wall. A regulary updated file accessible from `http://independencenumber.files.wordpress.com/2012/08/wow-july2004.pdf`. Available by request from S. Fajtlowicz at `siemion@math.uh.edu`.

[15] Fajtlowicz, S. On Conjectures of Graffiti – II. *Congressus Numerantium 60* (1987), 187 – 197.

[16] Fajtlowicz, S. On Conjectures of Graffiti – III. *Congressus Numerantium 66* (1988), 23 – 32.

[17] Fajtlowicz, S. On Conjectures of Graffiti – IV. *Congressus Numerantium 70* (1990), 231 – 240.

[18] Fajtlowicz, S. On Conjectures of Graffiti – V. In *Seventh International Quadrennial Conference on Graph Theory* (1995), vol. 1, pp. 367 – 376.

[19] Galinier, P., and Hao, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization 3*, 4 (1999), 379–397.

[20] Garey, M.R.., and Johnson, D.S.. *Computers and intractability. A guide to the theory of NP-completeness*. Freeman and Company, 1979.

[21] Goldberg, D., and Miller, B. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems 8* (1995), 193–212.

[22] Hakimi, S. On realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial and Applied Mathematics 10* (1962), 496–506.

[23] Hansen, P., Aouchiche, M., Caporossi, G., Mélot, H., and Stevanović, D. What Forms Do Interesting Conjectures Have in Graph Theory? In *Graphs and Discovery*, Fajtlowicz, S. *et al.*, Ed., vol. 69 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, 2005, pp. 231 – 252.

[24] Hansen, P., Hertz, A., Kilani, R., Marcotte, O., and Schindl, D. Average distance and maximum induced forest. *Journal of Graph Theory 60* (2009), 31–54.

[25] Hansen, P., and Mélot, H. Computers and Discovery in Algebraic Graph Theory. *Linear Algebra and its Applications 356* (2002), 211 – 230.

[26] Hansen, P., and Mélot, H. Variable Neighborhood Search for Extremal Graphs 9. Bounding the Irregularity of a Graph. In *Graphs and Discovery*, Fajtlowicz, S. *et al.*, Ed., vol. 69 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, 2005, pp. 253 – 264.

[27] Hansen, P., and Mladenović, N. Variable Neighborhood Search : Principles and Applications. *European Journal of Operational Research 130* (2001), 449 – 467.

[28] Havel, V. A remark on the existence of finite graphs. *Časopis pro Pěstování matematiky 80* (1955), 477–480.

[29] McKay, B.D. Nauty User's Guide (version 1.5). Tech. rep., Department of Computer Science, Australian National University, 1990.

[30] McKay, B.D. Isomorph-free Exhaustive Generation. *Journal of Algorithms 26* (1998), 306 – 324.

[31] Mélot, H. Facet defining inequalities among graph invariants: the system GraPHedron. *Discrete Applied Mathematics 156* (2008), 1875 – 1891.

[32] Mladenović, N., and Hansen, P. Variable Neighbourhood Search. *Computers and Operations Research 24* (1997), 1097 – 1100.

[33] Peeters, A., Coolsaet, K., Brinkmann, G., Van Cleemput, N., and Fack, V. GrInvIn in a nutshell. *Journal of Mathematical Chemistry 45* (2009), 471 – 477.

[34] Robertson, N., Sanders, D., Seymour, P., and Thomas, R. The Four-Color Theorem. *Journal of Combinatorial Theory, Series B 70* (1997), 2 – 44.

[35] Stevanović, D. and Branko, V. An Invitation to newGRAPH. In *Rendiconti del Seminario Matematico di Messina*, vol. 9 of *2*. 2003, pp. 211 – 216.

[36] Talbi, E.-G., Ed. *Metaheuristics - From design to implementation*. Wiley, Hoboken, 2009.

[37] Turán, P. Eine Extremalaufgabe aus der Graphentheorie. *Matematikai és Fizikai Lapok 48* (1941), 436–452.